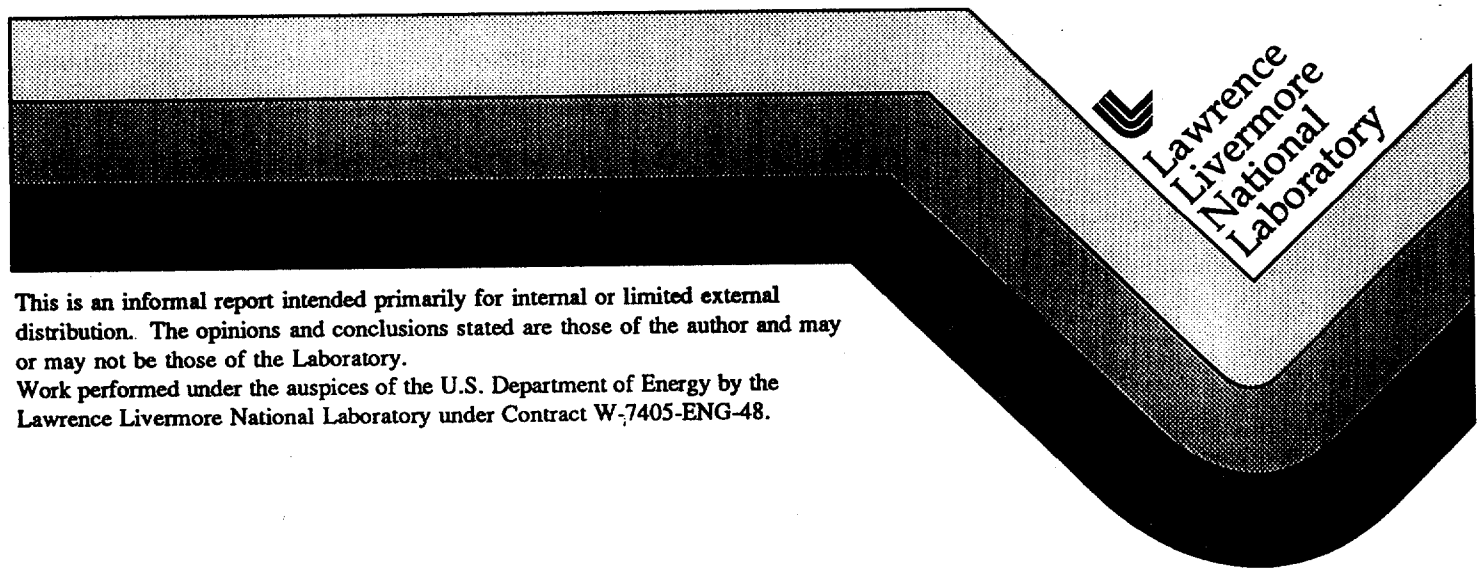
230729

UCRL-ID-117471

Single Event Upset Tests of a RISC-Based Fault-Tolerant Computer

J. R. Kimbrough, D. N. Butner, N. J. Colella, J. L. Kaschmitter,
D. L. Shaeffer, C. L. McKnett, P. G. Coakley, C. Casteneda

March 23, 1994



Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

Single Event Upset Tests of a RISC-Based Fault-Tolerant Computer

J. R. Kimbrough, D. N. Butner, N. J. Colella, J. L. Kaschmitter, D. L. Shaeffer

Lawrence Livermore National Laboratory*

Livermore, CA 94550

C. L. McKnett, P. G. Coakley

JAYCOR, Inc.

Santa Monica, CA 92121

C. Casteneda

Crocker Nuclear Laboratory

Davis, CA 95616

ABSTRACT

We developed a Single Event Upset (SEU) tolerant computer based on the MIPS, Inc., R3000A processor and commercial parts. Computer based SEU simulation followed up with proton SEU tests verified fault-tolerant operation. SEU tests demonstrated the capability of using SEU susceptible parts in Low Earth Orbit, with upset mitigation through dual-lock-stepped processors and software design methodologies.

INTRODUCTION

Spacecraft fault-tolerant computing is implemented in hardware and software at the part and system levels[1,2]. Ideally the additional hardware and software required for fault-tolerant features should have minimal impact on the computer system's size, speed, power requirements, and no decrease in computational performance. However, the part level approach often has an adverse impact on computational performance. The limitation on computational performance will become more significant as spacecraft computational requirements increase. A fundamental problem is that radiation hardened parts trail both the performance and the performance cost ratio of commercial parts due to long design cycle and small market for hardened parts. Concerns on SEU and Single Event Latchup (SEL) reliability issues of commercial parts may be solved by incorporating system level latchup mitigation and fault-tolerant computing technologies. System level latchup mitigation prevents burnout of devices due to single event current latchup.

Fault-tolerant software methods include N-version programming, recovery block, rollback and exception handling. These methods are not as robust as hardware solutions but provide a degree of fault-tolerant computing. N-version programming uses different programs to perform a task and compares their results. In recovery block approach, if the main program results for a given task fail an acceptability test, an alternate program version for the task is executed and checked. Rollback recovery is a backward error recovery technique implemented by resetting the program to the most recent checkpoint. Exception handling notes that an error occurred and attempts recovery.

Common hardware approaches to fault-tolerant computing include replication (three or more processors) with voting, duplication of processors with lock-step comparison, and a single processor with self-checking and watch dog timers. In replication, the results of the processors are compared and the majority result is considered correct. Under lock-step comparison, two processors are running in sync and the outputs continuously compared. When a difference in

outputs is detected the operation is halted and recovery attempted. In self-checking, the processor has built-in tests to check its operation. The watch dog timer approach requires the processor to periodically reset a timer, or else the timer resets the processor.

Our fault-tolerant computer combines software rollback recovery and duplicate processors with lock-step comparison [3]. Write buffers ensure that no data are written to memory once an error is detected and error detection and correction (EDAC) is applied to memory reads. The combination of write buffers and EDAC prevent corrupted data going to or from main memory. This requires comparing the output of the lock-stepped processors every cycle for possible error. When an error occurs, hardware causes a soft reset to occur. All data within the processor are assumed bad, while main memory is unaffected. The software is written with checkpoints, and upon restart will rollback to the most recent correctly written checkpoint. The soft restart operation takes about 10 msec, so the performance degradation, even in high SEU environments, is negligible.

HARDWARE

Initial developments involved the clock synchronization, the compare circuit, and the reset circuit to link two R3000 processors running on commercially available development boards. This allowed software development for error recovery and test while the fault-tolerant processor board was designed and built.

The fault-tolerant processor board consists of two sets of R3000A/R3010 RISC (Reduced Instruction Set Computer) processors with memory compare, reset circuitry and clock synchronization circuitry shown in Figure 1.

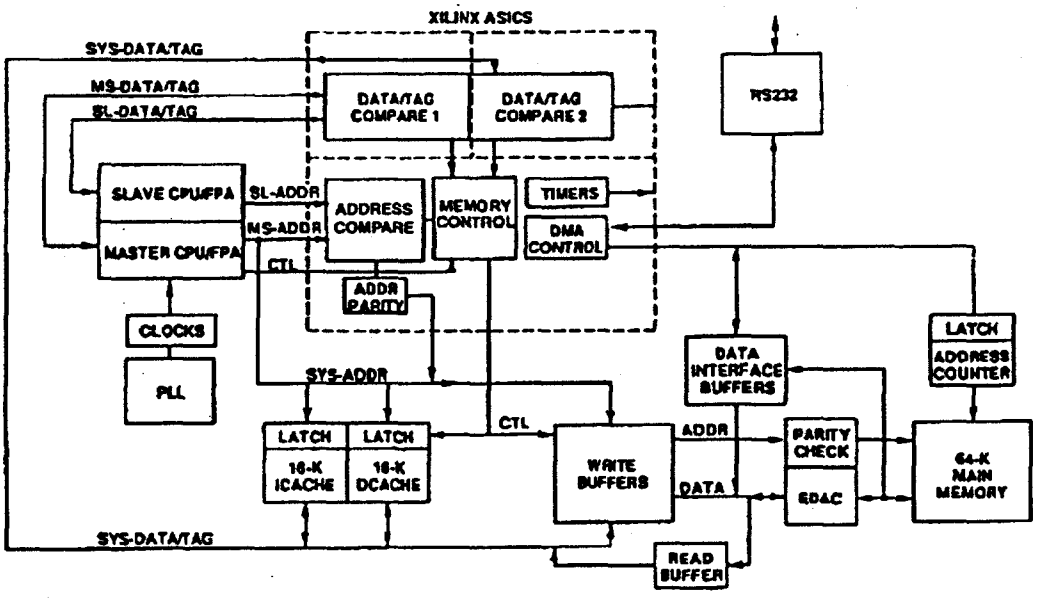


Figure 1. Block diagram of the computer.

The Xilinx 4000-50 series Field Programmable Gate Array (FPGA) contains the output compare and reset logic. Fast on and off chip delays allow the board to operate at over 16 MHz. The FPGA compares the address, data and TAG (cache addressing and status) from the two processors for errors. The FPGA also checks for address and data errors between the write buffer and main memory. Any error causes a CPU reset. In addition a TAG error clears and invalidates the caches and write buffer

Memory consists of instruction cache, data cache and main memory. The data and instruction caches are each 16 kwords with parity. The main memory is 64 kwords with parity and 7 check bits. The 7 check bits allows double bit error detection and single bit correction. A four word deep write buffer between the processors and main memory insures no bad processor data is written to main memory.

Communication is by a 9600 baud RS232 serial interface. Test programs are down loaded in to cache or main memory. The interface allows direct read access to main memory including check bits without EDAC. The direct read access allows checking for SEUs.

The phase lock loop circuit shown in Figure 2 adjusts the input clocks to both processors to achieve clock synchronization. The skew between the two SYSOUT clocks shown in Figure 3 is less than 400 picoseconds over a clock frequency range of 15 MHz to 33 MHz and a temperature range of 3° C to 67° C. Temperature measurements of skew were limited to the temperature range of commercial parts.

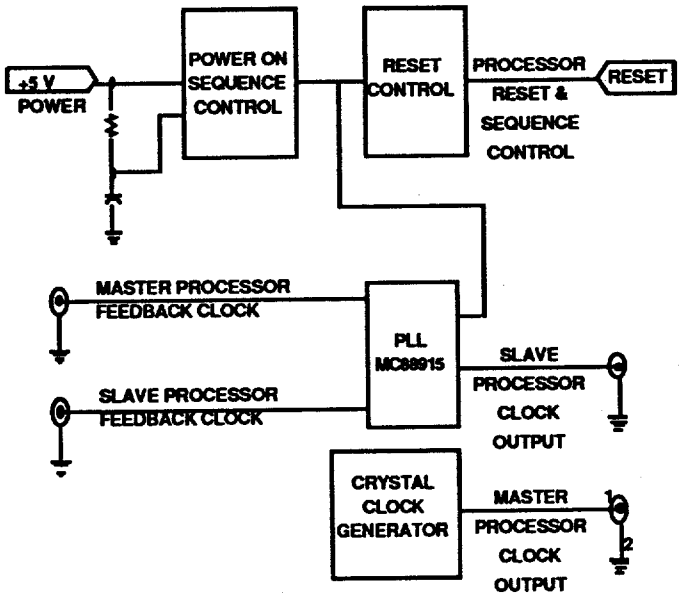


Figure 2. Clock synchronization circuit.

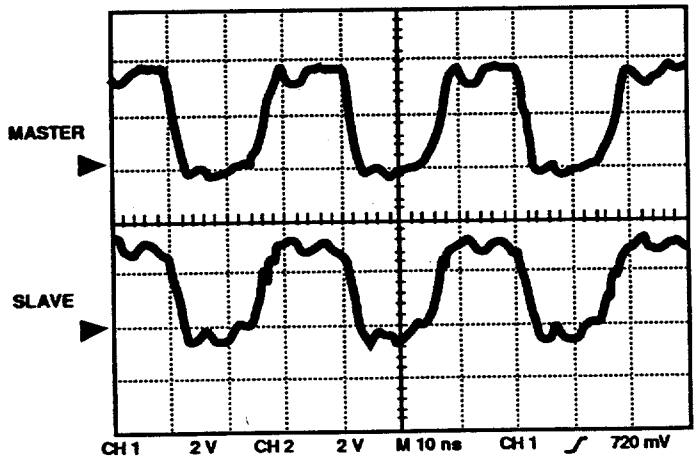


Figure 3. Master and Slave clocks with negligible skew.

The processor test board is 30 cm X 40 cm. This permits the processor under test to be isolated at least one collimated beam radius from the other components, including the R3020 write buffers, the Xilinx FPGA and the memories. The processor components are socket mounted so that different R3000A CPUs from different manufacturers can be tested.

SOFTWARE

Software recovery requires the processor to meet the following conditions: (1) Detect an error and prevent it from propagating to memory. (2) Restart within a few milliseconds from a previously saved state. (3) Memory containing critical data during the restart must be intact. We have designed, written and tested a "NOFAULT" software program that performs this function.

NOFAULT operates as four high level interconnected state machines. Each state machine saves and retrieves its state information using a set of routines that guarantees consistent state information is saved and retrieved. Each state machine normally performs an operation to completion before relinquishing control. The state machines may be interrupted the preemptive routine "PREEMPT." PREEMPT implements a state machine that always performs its operation to completion and communicates with one of the other state machines. The four high level state machines are "PREEMPT," "MASTER," "TEST1," and "TEST2." as shown in Figure 4.

The PREEMPT state machine is triggered by a timer interrupt. Once triggered it checks for data sent to it by the "MASTER" state machine. The data consists of x y coordinate pairs and an iteration number. Using the iteration number modulo 4, the PREEMPT state machine calculates another pair of x y coordinates and passes them back to the MASTER state machine. PREEMPT and MASTER use integer offsets to calculate the four corners of a square rotated by 45 degrees. The MASTER state machine checks the iteration and calculation. Any lost messages or failure of the PREEMPT routine to perform an operation to completion is detected. Communication to and from the PREEMPT routine is by two ring structures of predetermined sizes. One ring is for incoming data and the other for outgoing data.

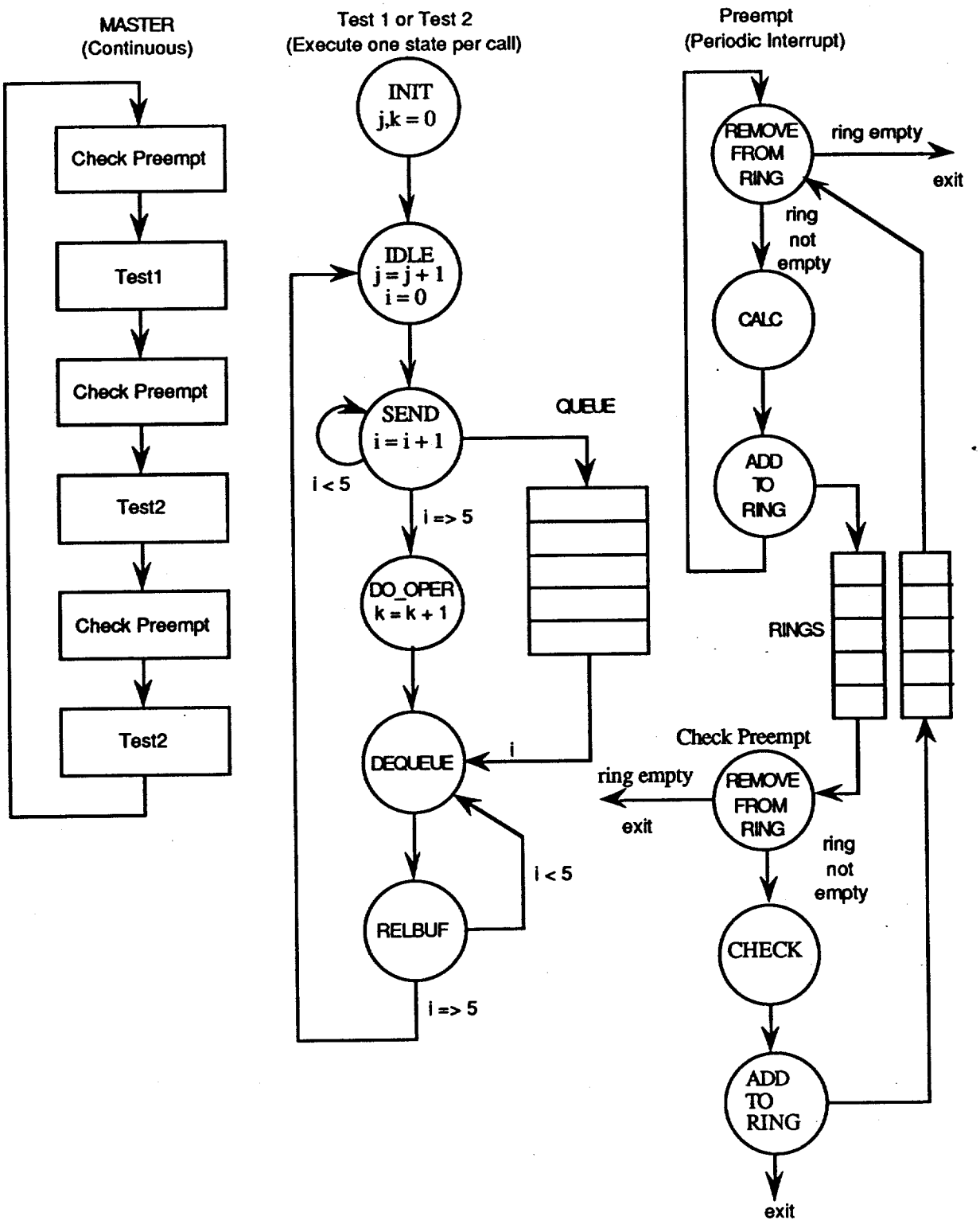


Figure 4. State diagram of the NOFAULT program.

The MASTER also keeps track of the program iterations and calls two test state machines, TEST1 and TEST2 which send data packets. TEST2 is called twice for every call to TEST1. TEST1 and TEST2 go through the following a sequence of states executing one state for each call from MASTER:

1. IDLE: increment the count and set the state to SEND.
2. SEND: send one data buffer (allocate and queue one data buffer), remember data buffer address, when five buffers have been sent, set the state to DO_OPER.
3. DO_OPER: increment the count and set the state to DEQUEUE.
4. DEQUEUE: dequeue the data buffer and check the address against the one remembered in SEND, then set the state to RELBUF.
5. RELBUF: release the data buffer memory, if five buffers have not been done, set the state to DEQUEUE, otherwise, set the state to IDLE.

The lower level communication routines are also implemented as state machines. The routines allocate and release memory from a memory buffer pool, to queue and dequeue data structures. These routines allow one state machine to allocate memory to hold a message to be passed to another state machine, to queue the message to the receiver, and to allow the receiver to handle the message and delete it. These routines eliminate the need for predetermined message sizes and queue lengths.

The implementation of these lower level routines is complicated. The routines modify data structures (the memory pool, queues, and rings) that may temporarily be in an inconsistent state. After determining if the requested operation can be performed, state information indicating the desired operation is saved, and the operation executed. Following execution state information indicating completion of the operation is saved. To guarantee continuity, the calling program saves the address of the state information in its state data structure before calling a routine again or completing its current operations.

The lower level routines also save status information in part of the caller's state data structure. This information is used to determine if the caller is calling the routine a second time due to a restart. The lower level routine then checks its internal state information to determine if the requested operation was performed. If the operation wasn't performed, the operation is performed and the required parameters are returned to the caller.

The test program keeps track in main memory of how many errors and recoveries occurred in the low level code sections and which of the four high level state machines called the low level code. In the low level routines errors can occur in the allocation, queuing, dequeuing, and releasing of data blocks.

PROTON SEU TESTS

Prior to the radiation effects tests the NOFAULT program underwent simulation testing to verify software recovery. Simulation testing done on a VAX used pseudo-random timer interrupts and routines to unwind the stack. Additional tests on a UNIX system used timer interrupts and software jump facilities. These two systems simulated recovery from both random events and events timed to cause upset during critical routines, such as during soft reboot.

The test application program utilizes cache memory for the stack and executes a variety of operations designed to detect error conditions. Each high level state machine data structure includes locations for counting detected errors and recoveries. In addition, the data structures associated with the lower level routines include locations for counting the number of times the routine executes its recovery code. The error and restart counts are monitored by having the program periodically print the numbers or by independently monitoring the memory of interest.

SEU tests were performed at the variable energy isochronous cyclotron housed at the Crocker Nuclear Laboratory at the University of California at Davis. The beam line is dedicated to proton SEU experiments and equipped with an automated beam current monitoring system. Current monitoring is done with a Faraday cup and a secondary electron emission monitor [4].

Our tests were conducted with 60 MeV protons, and a 3.5 inch thick Delrin block was used to collimate the beam onto the slave processor.

The proton SEU tests were done at fluxes in the range of 10^8 to 10^9 protons/cm^2-s. This flux range gave an approximate error rate ranging from one every ten seconds to once a second. In comparison, the peak flux of protons in the South Atlantic Anomaly is about 10^3 protons/cm^2-s. However, since the SEU rate is proportional to the flux, the SEU measurements are made at a higher flux in order to limit the time and expense of the testing. To ensure that there was no problem scaling laboratory fluxes to LEO fluxes, we collected upset data for Performance Semiconductor Inc., R3000s over four orders of magnitude of fluxes for 60 MeV protons. Figure 5 shows the device cross section versus flux and least squared fit. The data in Figure 3 indicate the SEU measurements are essentially independent of proton flux, and therefore the laboratory data may be used to estimate errors in space [5].

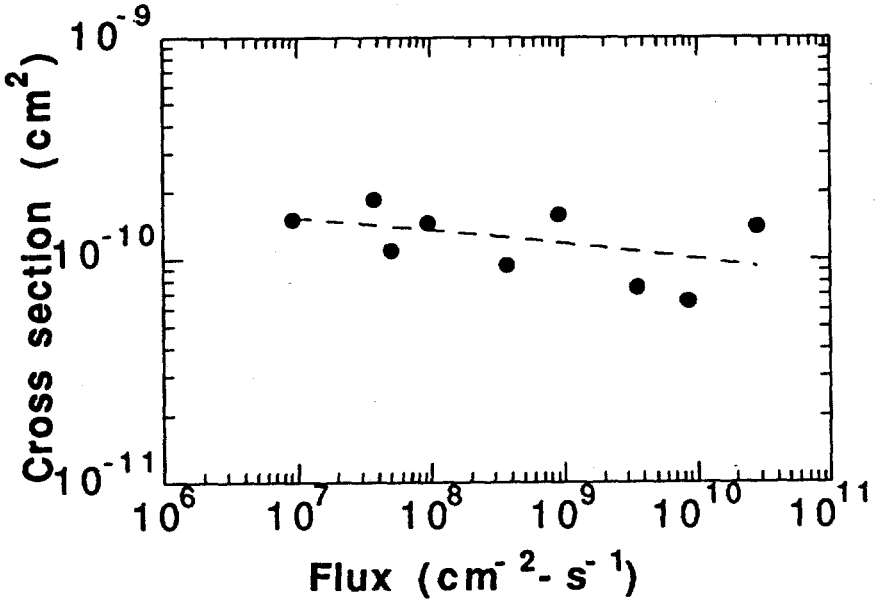


Figure 5. SEU cross section versus 60 MeV proton flux.

The test program ran on Integrated Device Technology, Inc. (IDT), 79R3000AE-25G144 9128CP processors, with the conditions and results shown in Table 1. In the first three experiments at a flux of $2x10^9$ protons/cm^2-s the program generated 4 errors and recoveries. The program quit communicating with the monitoring computer nearly as soon as the beam came on and resumed communication when the beam went off. However the program did record events and recoveries. The flux was reduced to $1.41x10^8$ protons/cm^2-s and the program successfully ran, demonstrating upset and recovery. In a series of tests, the flux was increased up to $3.31x10^9$ with 12 successful errors and recoveries. Ten of the successful errors and recoveries occurred during critical code sections in which data structures are potentially in inconsistent states. In addition, one error occurred during a critical code section handling preemptive data. During the last test, the program appeared to stop running and failed to recover when the beam was off. Later analysis of the data showed it was continuously executing a portion of the restart sequence getting an error and restarting. Prior to this upset in the restart sequence one error occurred in the handling of preemptive routine data. This last non recoverable problem appears to due to the combination of the two errors.

Table 1. Proton Test Data

Total Dose [krad(Si)]	Flux (P/cm^2-s)	Fluence (P/cm^2)	Cross-Section (cm^2/device)	Events
11.7	$2.0x10^9$	$8.4x10^{10}$		2
14.7	$2.0x10^9$	$2.1x10^{10}$		1
34.8	$2.3x10^9$	$1.4x10^{11}$		1
36.6	$1.4x10^8$	$1.3x10^{10}$	$7.7x10^{-11}$	1
71.9	$9.0x10^8$	$2.5x10^{11}$	$1.6x10^{-11}$	4
128.5	$1.8x10^9$	$4.1x10^{11}$	$4.9x10^{-12}$	2
280.3	$3.3x10^9$	$1.1x10^{12}$	$4.6x10^{-12}$	5*

* Hung after 5th event and recovery

The SEU device cross section based on the test program is 6.82×10^{-12} cm^2 $\pm$ 1.99×10^{-12} cm^2. Comparing this cross section to previous LLNL experiments shows the cross section is smaller than the "Stress Test" device cross section of 1.36×10^{-11} cm^2, and close to the "CPU Test" device cross section of 6.51×10^{-12} cm^2 [6]. This is expected since the "Stress Test" is a very extensive processor test that checks all 32 general registers and Table Lookaside Buffer. The "CPU Test" is more limited and only checks 6 of the general purpose registers and none of the Table Lookaside Buffer similar to a typical application program. The upset cross section is useful for estimating upset rate. However, the actual upset rate is dependent upon the application code.

PREDICTED PERFORMANCE IN ORBIT

SEU and SEL predictions due to protons and ions for a spacecraft with 60 mils of aluminum shielding in a 60 degree inclination 500 km orbit were calculated using "SPACERAD"[7]. Both Galactic Cosmic Ray and trapped proton environments use the IGRF 85 magnetic field model. The trapped proton environment is solar minimum (Sawyer & Vette) while the Galactic Cosmic Ray environment is Adam's solar minimum. The proton upset rate for this orbit is based on the Bendel and Petersen "A" parameter for the processor [8].

Prior proton SEU tests performed for single IDT R3000A operation produced an upset cross section of 1.4×10^{-11} cm^2 and implies an orbit upset rate of 5.7×10^{-5} upsets per day [6]. The unrecoverable upset cross section is much lower for the dual lock step configuration. Using the data from the last 4 runs that produced 12 successful recoveries for each potential upset, the upper bound estimate of the unrecoverable upset cross section is 5.7×10^{-13} cm^2 (one unrecoverable upset in the total fluence of 1.76×10^{12} p/cm^2). This upper bound estimate implies an orbit upset rate of 1.5×10^{-6} upsets per day.

The limiting factor testing and operation of the fault-tolerant computer is no longer the upset rate but total dose effects. A 245 krad(Si) exposure was required in order to measure the unrecoverable upset cross section.

CONCLUSION

The project successfully demonstrated that dual lock-step comparison of commercial RISC processors is a viable fault-tolerant approach to handling SEU in space environment. The fault tolerant approach on orbit error rate was 38 times less than the single processor error rate. The random nature of the upsets and appearance in critical code section show it is essential to incorporate both hardware and software in the design and operation of fault-tolerant computers.

REFERENCES

[1] D. A. Rennels, "Architectures for Fault-Tolerant Spacecraft Computers," Proceedings of the IEEE, Vol. 66, No. 10, pp. 1255-1268, Oct., 1978.

[2] D. P. Siewiorek, "Architecture of Fault-Tolerant Computers: An Historical Perspective", Proc. of the IEEE, Vol. 79, No. 12, Dec., 1991.

[3] J. L. Kaschmitter, D. L. Shaeffer, N. J. Colella, C. L. McKnett and P. G. Coakley, "Operation of Commercial R3000 Processors in the Low Earth Orbit (LEO) Space Environment", IEEE Trans. on Nucl. Sci., Vol 38, No. 6, pp. 1415-1420, Dec., 1991.

[4] K. M. Murray, W. J. Stapor, and C. Casteneda, "Calibrated Charged Particle Radiation System with Precision Dosimetric Measurement and Control," *Nucl. Instr. Methods Phys. Res.*, **A281**, pp. 616-621, 1989.

[5] D. L. Shaeffer, J. R. Kimbrough, J. W. Wilburn, S. M. Denton, N. J. Colella, P. G. Coakley, C. Casteneda, "Proton -Induced SEU Dose Effects, and LEO Performance Predictions for the MIPS R3000 Microprocessor," IEEE Nucl. Sci., NS-39, No. 6, pp. 2309-2315, Dec., 1992.

[6] J. R. Kimbrough, N. J. Colella, S. M. Denton, D. L. Shaeffer, D. Shih, J. W. Wilburn, P. G. Coakley, C. Casteneda, R. Koga, D. A. Clark, J. L. Ullmann, "Single Event Effects

and Performance Predications for Space Applications of RISC Processors," UCRL-JC-114877 Aug., 17, 1993.

[7] J. R. Letaw, author, SPACE RADIATION code, Severn Communications Corp., Millersvile, MD 1990.

[8] W. L. Bendel, E. L. Petersen, " Proton Upsets in Orbit," *IEEE Trans. Nucl. Sci.*, NS-30, No. 6, app. 4481-4485, Dec., 1983